

# Pattern Recognition: Object-to-Vector Conversion

Lecture Notes

December 2, 2025

## 1 Motivation

- Transformer-based Neural Networks are the basis for modern AI tasks:
  - Language translation (e.g., DeepL)
  - Text generation (e.g., ChatGPT)
- Based on Vaswani et al. (2017): *Attention is All You Need*.
- Before applying attention, words must be converted into high-dimensional vectors.

## 2 Basics

Objects can be words, sentences, pictures, sequences of pictures.

Words include:

- Conventional words: “I”, “can”, “do”, “it”
- Special tokens: SOS (Start of Sentence), EOS (End of Sentence)
- Punctuation marks

### 2.1 Distance Measures

[Euclidean Distance] For two vectors  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ :

$$\|\mathbf{x} - \mathbf{y}\|^2 = \sum_{i=0}^{N-1} (x_i - y_i)^2$$

[Dot Product Similarity]

$$\mathbf{x}^\top \mathbf{y} = \sum_{i=0}^{N-1} x_i y_i$$

Large if vectors are similar.

### 3 Context-Word Probabilities

[Center word probability] Given a center word  $w_n$  and context words  $w_{n-k}, \dots, w_{n+k}$ :

$$\sum_{k=-K, k \neq 0}^{k=K} p(w_{n+k} | w_n) = \sum_{k=-K, k \neq 0}^{k=K} p(\mathbf{x}_{n,k} | \mathbf{x}_n)$$

[Overall context/text probability (global probability)] Given a center word  $w_n$  and context words  $w_{n-k}, \dots, w_{n+k}$ ,  $N$  = words in the text:

$$\sum_{n=0}^{N-1} \sum_{k=-K, k \neq 0}^{k=K} p(w_{n+k} | w_n) = \sum_{n=0}^{N-1} \sum_{k=-K, k \neq 0}^{k=K} p(\mathbf{x}_{n,k} | \mathbf{x}_n)$$

### 4 Word Representation Options

- **Option 1:** Integer encoding
- **Option 2:** One-hot encoding
- **Option 3:** Dense vectors of dimension  $M \ll N$

### 5 Cost Function

[Log-Likelihood Cost] For a text with words  $w_1, \dots, w_N$ :

$$J(\boldsymbol{\theta}) = \frac{1}{N} \frac{1}{2K+1} \sum_{n=1}^N \sum_{k=-K, k \neq 0}^{k=K} \log(p(\mathbf{x}_{i(n,k)} | \mathbf{x}_{i(n)}, \boldsymbol{\theta}))$$

## 6 Training: Gradient-Based Optimization

---

**Algorithm 1** Gradient Descent for Word Vectors

---

```
Initialize word vectors  $\mathbf{x}_{\text{context}}$ 
for each epoch do
  for each center word  $\mathbf{x}_{\text{center}}$  do
    Compute gradient  $\nabla J(\theta)$ 
    Update:  $\mathbf{x}_{\text{context}} \leftarrow \mathbf{x}_{\text{center}} - \eta \nabla J$ 
  end for
end for
```

---

## 7 Estimating Conditional Probabilities

$$P(w_c|w_n) = \frac{\exp(\mathbf{x}_{\text{context}}^\top \mathbf{x}_{\text{center}})}{\sum_{n=0}^{N-1} \exp(\mathbf{x}_{\text{context}}^\top \mathbf{x}_{\text{center}})}$$

This is the **softmax** function.

## 8 Applications

- Word embeddings (Word2Vec, GloVe)
- Input for Transformer models
- NLP tasks: translation, summarization, sentiment analysis